Efficient Computation of Cohesive Subgraphs in Uncertain Bipartite Graphs

Gengda Zhao†, Kai Wang†, Wenjie Zhang†, Xuemin Lin†, Ying Zhang‡, Yizhang He†
†University of New South Wales, ‡University of Technology Sydney
{gengda.zhao, kai.wang, wenjie.zhang, xuemin.lin, yizhang.he}@unsw.edu.au, ying.zhang@uts.edu.au

***Abstract*—Bipartite graphs are extensively used to model relationships between two different types of entities. In many real-world bipartite graphs, relationships are naturally uncertain due to various reasons such as data noise, measurement error and imprecision of data, leading to uncertain bipartite graphs. In this paper, we propose the (α, β, η)-core model, which is the first cohesive subgraph model on uncertain bipartite graphs. To capture the uncertainty of relationships/edges, η-degree is adopted to measure the vertex engagement level, which is the largest integer k such that the probability of a vertex having at least k neighbors is not less than η. Given degree constraints α and β, and a probability threshold η, the (α, β, η)-core requires that each vertex on the upper or lower level have η-degree no less than α or β, respectively. An (α, β, η)-core can be derived by iteratively removing a vertex with η-degree below the degree constraint and updating the η-degrees of its neighbors. This incurs prohibitively high cost due to the η-degree computation and updating, and is not scalable to large bipartite graphs. This motivates us to develop index-based approaches. We propose a basic full index that stores (α, β, η)-core for all possible α, β, and η combinations, thus supporting optimal retrieval of the vertices in any (α, β, η)-core. Due to its long construction time and high space complexity, we further propose a probability-aware index to achieve a balance between time and space costs. To efficiently build the probability-aware index, we design a bottom-up index construction algorithm and a top-down index construction algorithm. Extensive experiments are conducted on real-world datasets with generated edge probabilities under different distributions, which show that (1) (α, β, η)-core is an effective model; (2) index construction and query processing are significantly sped up by the proposed techniques.**

I. Introduction

Bipartite graphs are a natural fit for modeling the interactions between two distinct types of entities, such as user-page networks [1], disease-symptom networks [2], [3], and protein-interaction networks [4]. In many real-world applications, interactions between entities are uncertain in nature [5], leading to uncertain bipartite graphs, where each edge occurs with a certain probability (i.e., existence probability). For instance, in a disease-symptom network, each edge indicates that a symptom associated with a disease occurs with a probability. For Covid-19, the probability that a patient will have a fever is 80%, and the probability for myocarditis is 1%. Cohesive subgraph mining is a fundamental research topic in graph analysis, which aims to find closely connected subgraphs wherein vertices are highly engaged. Most existing cohesive subgraph models on uncertain graphs are variants of those on unipartite graphs [6]–[13]. For example, the (k, η)-core [6]–[8] model is extended from k-core. While k-core only models vertex engagement as degrees in a deterministic graph, the η-degree in (k, η)-core can capture vertex engagement in an uncertain graph where edges are associated with probability values. On bipartite graphs, many cohesive subgraph models are also studied, such as (α, β)-core [14], [15], k-bitruss [16], [17], and all variants of bicliques [18], [19]. Among them, (α, β)-core is a widely used model that requires the vertices on the upper (lower) level to have at least α (β) neighbors. For instance, in Fig. 1, $\{u_1, u_2, u_3, u_4, v_2, v_3, v_4, v_5\}$ forms a $(3, 2)$-core. However, none of the existing cohesive models on bipartite graphs take the uncertainty of edges into account.

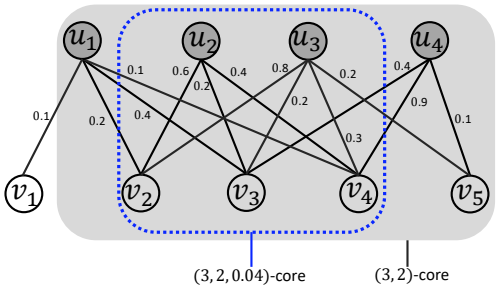


Fig. 1: Illustrating the (α, β, η)-core

To fill this research gap, in this paper, we propose the (α, β, η)-core model, which is the first cohesive subgraph model on uncertain bipartite graphs. Given an uncertain bipartite graph G and a probability threshold η, we adopt the η-degree from (k, η)-core to model the engagement of a vertex u, which is the largest integer k such that the probability that u has at least k neighbors is not less than η. Given degree constraints α, β, the (α, β, η)-core is the maximal subgraph of G such that the η-degree of each upper or lower vertex in the subgraph is no less than α or β, respectively. The (α, β, η)-core model can capture unique structures in uncertain bipartite graphs because it not only ensures the structural cohesiveness of the subgraph but also captures the probability information of edges. As shown in Fig. 1, the subgraph in shade is the $(3, 2)$-core. Since it directly imposes degree constraints on the deterministic graph, it includes all vertices except v_1. If we set $\alpha = 3$, $\beta = 2$, and $\eta = 0.04$, u_1, u_4, and v_5 will be excluded from the (α, β, η)-core because their η-degrees violate the degree constraints. The subgraph surrounded by blue dots is the $(3, 2, 0.04)$-core, which contains $\{u_2, u_3, v_2, v_3, v_4\}$. Note that it cannot be captured by any (α, β)-core.

Applications. Studying the (α, β, η)-core model has many applications. We list some of them below.

Categorisation of new diseases. The relationships between diseases and symptoms can be naturally modeled as an uncertain bipartite graph, where each edge indicates that a

disease is associated with a symptom with a certain probability [2], [3]. When a new disease (e.g., a different strain of the coronavirus) is spreading, the probability distribution of its common symptoms can be easily estimated from the data of new patients [20]. We can augment the existing disease-symptom uncertain bipartite graph by adding the edges and probabilities of the new disease and compute the (α, β, η)-core containing it. The (α, β, η)-core groups some existing diseases (e.g., SARS and influenza) and the new disease based on their shared and likely symptoms. These existing diseases in the same (α, β, η)-core are vital to understanding the new disease and devising effective pandemic prevention plans.

Fraud detection. On social media like Twitter, Facebook, and Instagram, interactions between users and pages form a bipartite graph [17], [21]. Each edge in this graph can be associated with a probability with which a user likes a page. Such probabilities can usually be estimated using machine learning models based on past user behaviors [22], [23]. Some fraudulent users give fake "likes" to the pages that they try to promote. The probability associated with these users and pages are significantly higher than average. Also, due to the advancement of anomaly detection, these users cannot rely on too many accounts for their activities [24]. Thus, these closely connected groups formed by fraudsters and the pages they promote can be captured by the (α, β, η)-core model.

Task-driven Team formation. For large corporations and universities to bid for a multidisciplinary research project, it is essential to find a team of researchers with areas of expertise relevant to a prescribed task [25]. To form such a team, it is often worthwhile to identify the expertise profile of researchers by investigating the track records in publication and research projects. The interactions between researchers and expertise form an uncertain bipartite graph, where each edge indicates how likely a researcher is to be considered as an expert for that area. These probabilities can be easily estimated based on past publications and successful grants. For the prescribed task, the (α, β, η)-core containing the most relevant expertise and researchers can be used as a candidate list to build the research team.

Motivations and challenges. To compute the (α, β, η)-core, a straightforward method is to start from the input uncertain bipartite graph and iteratively remove the vertices without enough η-degrees. However, this online computation approach fails inevitably when the input graph gets large, and it cannot support high volumes of (α, β, η)-core queries with different combinations of α, β and η values. Thus, we resort to indexing-based methods in this paper.

Existing works on computing (k, η)-core on uncertain unipartite graphs build an index to store the maximum η value such that a vertex is contained in (k, η)-core [6]. The index has the following features. First, the index supports optimal retrieval of vertices for given k and η values. Second, the index size is linear to the size of the graph (i.e., the number of edges). Likewise, to support (α, β, η)-core queries over uncertain bipartite graphs, we can store the maximum η value such that u is contained in (α, β, η)-core (i.e., the η-threshold of u) in a basic index. In this way, this index can support querying the vertices in arbitrary (α, β, η)-core by returning the vertices in (α, β)-core with η-thresholds no less than η. Such an index can also support retrieval of (α, β, η)-core vertices in optimal time. Nevertheless, the two-layer feature of bipartite graphs imposes extra challenges, and such a basic index suffers from large space complexity and long build time due to the large number of combinations of α and β. When trying to make the indexing method practically applicable, we face the following challenges.

1) *Challenge 1.* Designing an index with well-bounded space complexity that supports efficient querying of (α, β, η)-core is challenging.
2) *Challenge 2.* It is difficult to construct such an index efficiently due to the large number of combinations of α, β, and η that correspond to many (α, β, η)-cores.

Our solutions. To address Challenge 1, we propose the probability-aware index by only storing the vertices in the (α, β, η)-cores for some representative η values. For each selected η, we build a sub-index (i.e., a η-index) that stores the upper and lower vertices in the (α, β, η)-cores separately in a space-compact manner as in [15]. For an incoming (α, β, η)-core query, if the η-index is built, the query can be responded to in optimal time. Otherwise, we find the closest two η values in the index ($\eta_1 < \eta < \eta_2$) and use the vertices in (α, β, η_1)-core and (α, β, η_2)-core to compute (α, β, η)-core, which is much faster than computing (α, β, η)-core from the input graph.

A natural question is: which η values should we build an η-index for. We observe that the η values corresponding to different sub-indexes are not distributed evenly. Thus, if we choose a series of η values with fixed step size, it is very likely that adjacent η values correspond to identically built sub-indexes, which results in redundancy. It is worth noting that as η increases, the number of vertices in the (α, β, η)-cores that can be queried from the η-index (denoted as $\mathcal{T}(\eta)$) is decreasing. Since $\mathcal{T}(\eta)$ can reflect the ability of the η-index to answer queries, we gradually increase η such that $\mathcal{T}(\eta)$ approximately halve in size to reduce redundancy. This results in a well-bounded space complexity of the index (i.e., $O(m \cdot log(m))$, where m is the total number of edges in the input graph). As evaluated in the experiments, this probability-aware index is able to strike a balance between index space cost and query time.

To address Challenge 2, we propose efficient algorithms to construct the probability-aware index. For each selected η, we build an η-index that records which vertices are contained in the (α, β, η)-cores. Since the structure of each η-index resembles the index for (α, β)-core [15], we devise a similar bottom-up index construction algorithm that processes the (α, β, η)-cores with smaller degree constraints (α or β) first. However, in this approach, the vertices with high η-degrees can be updated many times, especially when α or β is small. This compromises the efficiency of index construction. To address this issue, we present a top-down index construction algorithm that prioritizes processing the (α, β, η)-cores with large degree constraints, which contain many vertices with high η-degrees. Immediately, these vertices with high η-degrees in the (α, β, η)-cores with large α or β will occur in the (α, β, η)-cores with smaller α or β as well. Thus, the

η-degrees of these vertices need not to be updated as we visit the (α, β, η)-cores with smaller degree constraints. In this manner, the total number of η-degree updates is significantly decreased compared to the bottom-up approach, and the index construction is sped up noticeably.

Contribution. Here we summarize our principal contributions.

- We propose the first cohesive subgraph model, (α, β, η)-core, on uncertain bipartite graphs.
- We propose a probability-aware index with $O(m \cdot log(m))$ space complexity, which is much smaller in practice (e.g., its space usage is only 1.03×-2.22× of the graph size as evaluated in the experiments). In addition, it supports efficient query processing.
- We devise efficient index construction algorithms that can build the probability-aware index in bottom-up and top-down manners.
- We conduct extensive experiments on 10 real bipartite graphs with edge probabilities generated under different distributions to validate the effectiveness of our (α, β, η)-core model and the efficiency of the proposed algorithms.

Organization. The rest of the paper is organized as follows. Section II presents the basic definitions and the problem statement. Section III presents the basic full index. Section IV presents the probability-aware index and the index construction algorithms. Section V reports and analyzes the experimental results. Section VI reviews the related works, and Section VII concludes the paper.

II. Preliminaries

In this section, we formally introduce the notations and basic concepts for defining the (α, β, η)-core.

TABLE I: Summary of Notations

Notation	Definition
G	a deterministic bipartite graph
$\mathcal{G}$	an uncertain bipartite graph
$deg_\eta(u, \mathcal{G})$	the η-degree of u in $\mathcal{G}$
$thres_{\alpha,\beta}(u)$	the η-threshold of u in $\mathcal{G}$
α_{max}	the maximal α such that the $(\alpha, 1)$-core exists in $\mathcal{G}$
$\beta_{max}(\alpha)$	the maximal β such that the (α, β)-core exists in $\mathcal{G}$
$\mathcal{I}_f$	the full index of $\mathcal{G}$
$\mathcal{I}_{pa}$	the probability-aware index of $\mathcal{G}$
η-index	the sub-index with probability η in $\mathcal{I}_{pa}$
$\mathcal{T}(\eta)$	the number of vertices in the all (α, β, η)-cores
α-offset	the maximal β value where u in an (α, β, η)-core
β-offset	the maximal α value where u in an (α, β, η)-core

A. The definition of (α, β, η)-core

Our problem is defined over an uncertain bipartite graph $\mathcal{G}(V = (U, L), E, p)$. $V(\mathcal{G})$ is the set of vertices in $\mathcal{G}$, $U(\mathcal{G})$ represents the set of upper layer vertices, and $L(\mathcal{G})$ represents the set of lower layer vertices, where $U(\mathcal{G}) \cap L(\mathcal{G}) = \emptyset$, $V(\mathcal{G}) = U(\mathcal{G}) \cup L(\mathcal{G})$. $E(\mathcal{G}) \subseteq U(\mathcal{G}) \times L(\mathcal{G})$ denotes the set of edges in $\mathcal{G}$, and p is a function that maps the existence probability of each edge to a real value in $[0, 1]$. The probability of an edge $e \in E(\mathcal{G})$ is denoted by p_e. The degree of u in $\mathcal{G}$ is denoted as $deg(u, \mathcal{G})$. Assuming all the vertices and edges in $\mathcal{G}$ exist, we have a deterministic graph G^*. We use $N(u, \mathcal{G})$ to denote the neighbor set of a vertex u in G^*, and we use $Deg(u, \mathcal{G})$ to denote the degree of u in G^*. We use n and m to denote the number of vertices and edges in G^*, respectively. Note that $\mathcal{G}$ is omitted when the context is clear.

We first introduce the concept of (α, β)-core on deterministic bipartite graphs.

Definition 1. (α, β)**-core.** *Given a bipartite graph $G(V = (U, L), E)$, and two degree constraints α and β, the (α, β)-core is a subgraph G' which satisfies (1) $deg(u, G') \geq \alpha$ for each $u \in U(G')$ and $deg(v, G') \geq \beta$ for each $v \in L(G')$; (2) G' is maximal, i.e., any supergraph $G'' \supset G'$ is not an (α, β)-core.*

According to previous uncertain graph models and the well-known possible-world semantics [6]–[8], we assume the existence probability of each edge in $\mathcal{G}$ is independent. Based on this assumption, there exist 2^m possible graph instances by given an uncertain bipartite graph $\mathcal{G}$ with m edges. Let $G(V{=}(U, L), E)$ denote a graph instance. The probability of observing a graph instance G is denoted by $Pr(G)$, which can be calculated by:

$$Pr(G) = \prod_{e \in E(G)} p_e \prod_{e \in E(\mathcal{G}) \setminus E(G)} (1 - p_e). \tag{1}$$

Given an uncertain bipartite graph $\mathcal{G}$, let $\mathcal{G}_u^{\geq k}$ be the set of all possible graph instances where u has a degree of at least k, i.e., $\mathcal{G}_u^{\geq k} = \{G \subseteq \mathcal{G} \mid deg(u, G) \geq k\}$. Since different possible graph instances are independent [8], we have the following equation:

$$Pr[deg(u, \mathcal{G}) \geq k] = \sum_{G \in \mathcal{G}_u^{\geq k}} Pr(G). \tag{2}$$

According to the above discussions, we introduce the concepts of η-degree and (α, β, η)-core.

Definition 2. η**-degree.** *Given an uncertain bipartite graph $\mathcal{G}(V{=}(U, L), E, p)$, a vertex $u \in \mathcal{G}$, and a probabilistic threshold $\eta \in [0, 1]$, the η-degree of a vertex u, denoted by $deg_\eta(u, \mathcal{G})$, is defined as $deg_\eta(u, \mathcal{G}) = max\{k \mid Pr[deg(u, \mathcal{G}) \geq k] \geq \eta\}$.*

Definition 3. (α, β, η)**-core.** *Given an uncertain bipartite graph $\mathcal{G}(V{=}(U, L), E, p)$, two integers α, β and a probabilistic threshold $\eta \in [0, 1]$, a subgraph $\mathcal{G}'$ is the (α, β, η)-core of $\mathcal{G}$ if (1) $deg_\eta(u, \mathcal{G}') \geq \alpha$ for each $u \in U(\mathcal{G}')$ and $deg_\eta(u, \mathcal{G}') \geq \beta$ for each $v \in V(\mathcal{G}')$; (2) $\mathcal{G}'$ is maximal, i.e., any supergraph $\mathcal{G}'' \supset \mathcal{G}$ is not an (α, β, η)-core.*

Problem Statement. Given an uncertain bipartite graph $\mathcal{G}(V{=}(U, L), E, p)$, two integers α, β and a probabilistic threshold $\eta \in [0, 1]$, we study the problem of computing the vertices in the (α, β, η)-core of $\mathcal{G}$.

Example 1. *Consider the graph $\mathcal{G}$ in Figure 1. Given query parameters $\alpha = 3$, $\beta = 2$, and $\eta = 0.02$, the $(3, 2, 0.02)$-core of $\mathcal{G}$ contains $\{u_2, u_3, u_4, v_2, v_3, v_4, v_5\}$. Given query parameters $\alpha = 3$, $\beta = 2$, and $\eta = 0.04$, the $(3, 2, 0.04)$-core of G contains $\{u_2, u_3, v_2, v_3, v_4\}$.*

B. Warm up

To simplify the calculation of η-degree, Bonchi et al. [8] propose several equations as follows.

$$\begin{aligned} Pr[deg(u,\mathcal{G}) \geq k] &= \sum_{i \geq k} Pr[deg(u,\mathcal{G}) = i] \\ &= 1 - \sum_{i \leq k\text{-}1} Pr[deg(u,\mathcal{G}) = i]. \end{aligned} \tag{3}$$

Based on Equation 3, the problem of calculating η-degree is converted to computing $Pr[deg(u,\mathcal{G}) = i]$ ($i \leq k-1$). Let $s = Deg(u,\mathcal{G})$ and $E_u(\mathcal{G}) = \{e_1, e_2, ..., e_s\}$ be the set of edges that incident to u in $\mathcal{G}$. There are two possible situations: (i) there are i-1 edges exist in $\{e_1, e_2, ..., e_{s-1}\}$ and e_s exists; or (ii) there are i edges exist in $\{e_1, e_2, ..., e_{s-1}\}$ and e_s does not exist. Let $E_u^h(\mathcal{G}) = \{e_1, e_2, ..., e_h\}$ be a subset that contains the first h edges of $E_u(\mathcal{G})$, where $h \leq s$. Let $\mathcal{G}_u^h$ denote a subgraph of $\mathcal{G}$ that excludes the edges in $E_u(\mathcal{G}) \setminus E_u^h(\mathcal{G})$ (i.e., $\mathcal{G}_u^h = (V\text{=}(U,L), E \setminus (E_u(\mathcal{G}) \setminus E_u^h(\mathcal{G})), p)$). Let $X_u(h,j)$ be the probability of a vertex u that has a degree j in $\mathcal{G}_u^h$, $j \leq h \leq s$. Obviously, $X_u(s,j)$ is equal to $Pr[deg(u,\mathcal{G}) = j]$. Then we can get the dynamic programming recursive function of calculating $X_u(s,i)$ (i.e., $Pr[deg(u,\mathcal{G}) = i]$):

$$X_u(h,j) = p_{e_h} X_u(h\text{-}1, j\text{-}1) + (1\text{-}p_{e_h}) X_u(h\text{-}1, j). \tag{4}$$

The initialization case are: (i) $X_u(0,0) = 1$, (ii) for all $h \in [0,s]$, $X_u(h,-1) = 0$, (iii) for all $0 \leq h < j \leq s$, $X_u(h,j) = 0$. The time complexity for computing the η-degree of a vertex u is $O(Deg(u,\mathcal{G}) \cdot deg_\eta(u,\mathcal{G}))$.

[8] also proposes an updating technique to dynamically update the η-degree when an edge is removed. Given an edge $e = (u,v) \in \mathcal{G}$. After removing e, we get a subgraph $\mathcal{G}_{\neg e} = (V = (U,L), E \setminus e, p)$ of $\mathcal{G}$. The η-degree of u can be updated by computing $Pr[deg(u,\mathcal{G}_{\neg e}) = j]$, where $j \in [0, deg_\eta(u,\mathcal{G})]$. To simplify, $Pr[deg(u,\mathcal{G}_{\neg e}) = j]$ is denoted by $p(u,\neg e,j)$ and can be computed as follows.

$$p(u,\neg e,j) = \frac{Pr[deg(u,\mathcal{G} = j]\text{-}p_e(p(u,\neg e,j\text{-}1)}{1\text{-}p_e}. \tag{5}$$

Based on Equation 5 and the initial state $Pr[deg(u,\mathcal{G}_{\neg e}) = 0] = Pr[deg(u,\mathcal{G}) = 0]/(1\text{-}p_e)$, the time complexity of computing $Pr[deg(u,\mathcal{G}_{\neg e}) = j]$ for all $j \in [1, deg_\eta(u,\mathcal{G})]$ is $O(deg_\eta(u,\mathcal{G}))$ time.

III. Baseline Solutions

In this section, we introduce the online computation algorithm and the full index to retrieve (α,β,η)-core.

A. The online algorithm

In [26], the authors propose an online peeling algorithm to compute the (α,β)-core on deterministic bipartite graphs by iteratively removing the vertices that violate the degree constraints). Using a similar peeling idea, we introduce an online algorithm UC-Q$_o$ to compute the (α,β,η)-core in $\mathcal{G}$. We can directly have the following observation from Definition 1 and Definition 3.

Observation 1. *Given an uncertain bipartite graph $\mathcal{G}$, for any α,β, and η, (α,β,η)-core $\subseteq$ (α,β)-core.*

Based on Observation 1, we can compute the (α,β,η)-core from the (α,β)-core by iteratively removing the vertices without enough η-degree, as outlined in Algorithm 1. In Algorithm 1, UC-Q$_o$ first removes all the vertices do not belong to the (α,β)-core using the algorithm in [26]. After that, we calculate the η-degree of the remaining vertices using Equation 4 (Line 2). Then, we iteratively remove the upper (lower) vertices with η-degree less than α (β) (Lines 3-14). The time complexity of Line 1, Line 2, and Lines 3-16 is $O(m)$, $O(\sum_{u \in V} deg_\eta(u) \cdot Deg(u))$, $O(\sum_{u \in V} \sum_{v \in N(u)} deg_\eta(v))$, respectively. Thus, the total time complexity of Algorithm 1 is $O(\sum_{u \in V} deg_\eta(u) \cdot Deg(u) + \sum_{u \in V} \sum_{v \in N(u)} deg_\eta(v))$.

Algorithm 1: UC-Q$_o$

Input: An uncertain graph $\mathcal{G} = (v = (U,L), E, p)$, two integer α, β and a probability threshold $\eta \in [0,1]$
Output: the (α,β,η)-core in $\mathcal{G}$

1 remove all the vertices do not belong to the (α,β)-core;
2 compute $deg_\eta(u)$ for all $u \in V(\mathcal{G})$;
3 **repeat**
4 **while** $\exists u \in U(\mathcal{G})$ *s.t.* $deg_\eta(u) < \alpha$ **do**
5 **foreach** $v \in N(u)$ **do**
6 remove the edge (u,v) from $\mathcal{G}$;
7 update $deg_\eta(v)$;
8 $U(\mathcal{G}) \leftarrow U(\mathcal{G}) \setminus \{u\}$;
9 **while** $\exists v \in L(\mathcal{G})$ *s.t.* $deg_\eta(v) < \beta$ **do**
10 **foreach** $u \in N(v)$ **do**
11 remove the edge (u,v) from $\mathcal{G}$;
12 update $deg_\eta(u)$;
13 $L(\mathcal{G}) \leftarrow L(\mathcal{G}) \setminus \{v\}$;
14 **until** *$U(\mathcal{G})$ and $L(\mathcal{G})$ do not change*;
15 **return** $V(\mathcal{G})$

B. The full index

Since the online computation algorithm needs to peel the graph from scratch, it is time-consuming in many circumstances, especially when the bipartite graph is large. To support efficient query processing, a general idea is to use index-based approaches. In this subsection, we introduce the full index $\mathcal{I}_f$ that can support retrieving the vertices in an arbitrary (α,β,η)-core in optimal time. We can have the following lemma based on the nested structure of (α,β,η)-core.

Lemma 1. *Given parameters α, β, η, and η', (α,β,η)-core $\subseteq$ (α,β,η')-core if $\eta > \eta'$.*

Proof. This is immediate from Definition 3. □

According to the above lemma, we only need to concern the largest probability η for each vertex that can be in the (α,β,η)-core. We call such η value the η-threshold, which is formally defined as follows.

Definition 4. η-threshold. *Given an uncertain bipartite graph $\mathcal{G}(V\text{=}(U,L), E, p)$, two integers α,β, and a vertex u, the largest η such that an (α,β,η)-core containing u exists is called the η-threshold of u, denoted by $thres_{(\alpha,\beta)}(u)$.*

For each α and β combination, we can store the vertices with non-zero η-thresholds along with their η-threshold values

into the index. Then, when querying an (α, β, η)-core, we can just return each vertex u with $thres_{(\alpha,\beta)}(u) \geq \eta$. Specifically, $\mathcal{I}_f$ is a three-level index. The first and second levels of $\mathcal{I}_f$ are arrays of pointers for α/β values. The third level of $\mathcal{I}_f$ contains a list of vertex blocks. Each vertex block vb is associated with a threshold $vb.\eta$ and contains the set of vertices $\{u \in V(\mathcal{G}) \mid thres_{(\alpha,\beta)}(u) = vb.\eta\}$. We show an example of this full index 2 as follows.

Example 2. *Consider the uncertain bipartite graph $\mathcal{G}$ in Figure 1. We show the corresponding full index $\mathcal{I}_f$ in Figure 2. In the index, all the pre-computed η-thresholds of all vertices are stored and shown in the vertex blocks of the bottom level. For example, $(1, 3, 0.108)$-core is $\{u_2, u_3, u_4, v_4\}$ and it has the largest η-threshold among all $(1, 3, \eta)$-cores. Thus, the vertex block with $\{u_2, u_3, u_4, v_4\}$ and the η-threshold 0.108 is pointed by the pointer in the third element of the first array in β level. Because the η-threshold of u_1 in $(1, 3, \eta)$-core is 0.1, u_1 is stored in a different vertex block.*

Query based on $\mathcal{I}_f$. The query processing with the index $\mathcal{I}_f$ is shown in Algorithm 2. Given query parameters α, β, and η, if either $\mathcal{I}_f[\alpha]$ or $\mathcal{I}_f[\alpha][\beta]$ is empty, it returns $\emptyset$ as the result since the (α, β, η)-core is empty (Lines 1-2). Otherwise, we retrieve the vertex block referred by the pointer $\mathcal{I}_f[\alpha][\beta]$ (Line 3). We iteratively get all the vertices in the blocks with threshold large than or equal to the given query value η. We can easily get that the time complexity of Algorithm 2 is $O(|C|)$, where C is the set of all the vertices in (α, β, η)-core. Note that Algorithm 2 is optimal, since its time complexity is bounded by the result size. For example, to query the $(2, 2, 0.12)$-core from $\mathcal{I}_f$, we simply visit the corresponding pointers in each level as shown in Figure 2. Then, we collect the vertices in the vertex blocks with η-thresholds greater than or equal to 0.12. The $(2, 2, 0.12)$-core includes vertices with η-thresholds 0.168 and 0.124, which is $\{u_1, u_2, u_3, u_4, v_2, v_3, v_4\}$.

Algorithm 2: Query based on $\mathcal{I}_f$

Input: An uncertain graph $\mathcal{G} = (V = (U, L), E, p)$, two integers α, β, a probabilistic threshold η, and $\mathcal{I}_f$
Output: the (α, β, η)-core in $\mathcal{G}$

1 **if** $\mathcal{I}_f.size() < \alpha$ *or* $\mathcal{I}_f[\alpha].size() < \beta$ **then**
2 **return** $\emptyset$;
3 $vb \leftarrow$ the vertex block pointed by $\mathcal{I}_f[\alpha][\beta]$;
4 $C \leftarrow \emptyset$;
5 **while** $vb.\eta \geq \eta$ **do**
6 **foreach** $u \in vb$ **do**
7 $C \leftarrow C \cup u$;
8 $vb \leftarrow$ the next vertex block in $\mathcal{I}_f[\alpha][\beta]$;
9 **return** C

Build the full index. Based on the structure of $\mathcal{I}_f$, it is easy to build the index if we know the η-threshold for each vertex regarding all possible α and β combinations. Thus, we focus on computing the η-threshold for each vertex $u \in V(\mathcal{G})$. We first give the following definition.

Definition 5. d-probability. *Given an uncertain bipartite graph $\mathcal{G}$ and two integers α, β, the d-probability $prob(u, \mathcal{G})$ of a vertex u is defined as follows. (1) if $u \in U(\mathcal{G})$, $prob(u, \mathcal{G})$ is the probability that $Pr[deg(u) \geq \alpha]$; (2) if $u \in L(\mathcal{G})$, $prob(u, \mathcal{G})$ is the probability that $Pr[deg(u) \geq \beta]$.*

Based on Definition 5, the details of the algorithm that computes the η-threshold of all the vertices regarding all possible α and β combinations are shown in Algorithm 3, which iteratively removes the vertex with the minimum d-probability.

Algorithm 3: η-threshold Computation

Input: An uncertain graph $\mathcal{G}(V$=$(U, L), E, p)$
Output: $thres_{(\alpha,\beta)}(u)$ for each $u \in V(\mathcal{G})$ regarding all possible α and β

1 **for** $\alpha = 1$ *to* α_{max} **do**
2 **for** $\beta = 1$ *to* $\beta_{max}(\alpha)$ **do**
3 $\mathcal{G}' \leftarrow$ the (α, β)-core in $\mathcal{G}$;
4 compute $prob(u, \mathcal{G}')$ for each $u \in V(\mathcal{G}')$;
5 **while** $\mathcal{G}'$ *is not empty* **do**
6 $\eta_{min} \leftarrow min_{v \in V(\mathcal{G}')} prob(v, \mathcal{G}')$;
7 $u \leftarrow$ the vertex with $prob(u, \mathcal{G}') = \eta_{min}$;
8 $thres_{(\alpha,\beta)}(u) \leftarrow \eta_{min}$;
9 $V(\mathcal{G}') \leftarrow V(\mathcal{G}') \backslash \{u\}$;
10 **foreach** $v \in N(u)$ **do**
11 update $prob(v, \mathcal{G}')$;
12 **return** $thres_{(\alpha,\beta)}(u)$ *for each* $u \in V(\mathcal{G})$ regarding all possible α and β;

In Algorithm 3, for each α and β combination, we first get the (α, β)-core $\mathcal{G}'$ from $\mathcal{G}$ (Line 3). Then, we compute the d-probability $prob(u, \mathcal{G}')$ for each vertex w.r.t. α and β (Line 4). The η-threshold for all vertices in $\mathcal{G}'$ under α and β are computed from Line 5 to Line 12. Specifically, we assign η_{min} as the minimum d-probability of the vertices in $\mathcal{G}'$ and get a vertex with the minimum d-probability in Lines 6-7. Then, we assign the η-threshold of u as η_{min} in Line 8. We then remove the vertex u and update the d-probability of u's neighbors in Lines 10-11. At last, we return all η-threshold of all vertices regarding all possible α and β in $\mathcal{G}$.

Complexity analysis. For each α and β combination, running Line 3 needs $O(m)$ time [26]. The time complexity of line 4 is $O(\sum_{u \in V} deg_\eta(u) \cdot Deg(u))$, as discussed in Section II. We use a min priority queue (implemented by heap) to maintain all the vertices, and the keys are their d-probabilities. Thus, Lines 6-9 totally take $O(n \cdot \log n)$ time to find and remove the vertex with the minimum d-probability. The time complexity of Lines 10-11 is $O(\sum_{u \in V} \sum_{v \in N(u)} deg_\eta(v))$. If we use Δ to represent the maximum η-degree over all the vertices, the time complexity of Line 4 and Lines 9-11 can be simplify to $O(m \cdot \Delta)$. Thus, the total time complexity of Algorithm 3 is $O(\sum_{\alpha=1}^{\alpha_{max}} \sum_{\beta=1}^{\beta_{max}(\alpha)} (n \cdot \log n + m \cdot \Delta))$.

Based on the result of Algorithm 3, we can easily build $\mathcal{I}_f$. We first put the vertices into vertex blocks based on their η-threshold and (α, β) values. After that, $\mathcal{I}_f[\alpha]$ stores the address of the α-th array in SPT. $\mathcal{I}_f[\alpha][\beta]$ saves the address of the first vertex block which has the largest η-threshold for a (α, β) pair. Constructing $\mathcal{I}_f$ from the result of Algorithm 3 can be done in $O(\sum_{\alpha=1}^{\alpha_{max}} \beta_{max}(\alpha) \cdot n)$ time, which is less than the time complexity of Algorithm 3. Thus, the total time complexity of constructing $\mathcal{I}_f$ is $O(\sum_{\alpha=1}^{\alpha_{max}} \sum_{\beta=1}^{\beta_{max}(\alpha)} (n \cdot \log n + m \cdot \Delta))$. In addition, since the size of vertex blocks for a pair of (α, β) is bounded by $O(n)$, $\mathcal{I}_f$ takes $O(\sum_{\alpha=1}^{\alpha_{max}} \cdot \beta_{max}(\alpha) \cdot n)$ space.

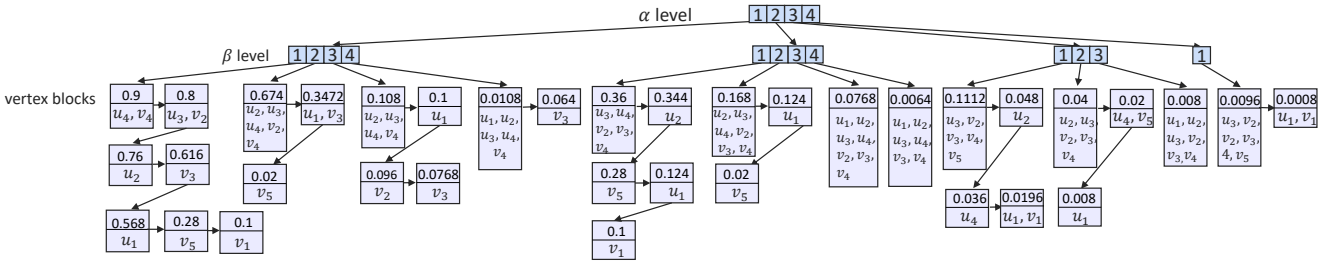


Fig. 2: Illustrating the full index $\mathcal{I}_f$ of $\mathcal{G}$ in Fig.1

IV. The probability-aware index

Motivation. Although the full index $\mathcal{I}_f$ can support optimal queries, it is not practical due to its high space complexity and time complexity as analyzed above. To address this issue, we propose a probability-aware index $\mathcal{I}_{pa}$ that strikes a balance between the indexing cost and query time. In Section IV-A, we show the intuition of designing $\mathcal{I}_{pa}$ and provide an overview of $\mathcal{I}_{pa}$. The query processing algorithm based on $\mathcal{I}_{pa}$ is presented in Section IV-B, and the index construction algorithms are introduced in Section IV-C.

A. Index overview

In [15], Liu et al. propose the bicore index to support retrieval of the vertices of an arbitrary (α, β)-core in optimal time, which needs $O(m)$ space.

It is intuitive that for a given η value, we can build an η-index to store the vertices in all (α, β, η)-cores (with the same η), whose structure is similar to the bicore index. In this way, the vertices in any (α, β, η)-core can be retrieved from the η-index in optimal time. Note that the size of the η-index drops as η increases. Since we cannot afford to build indexes for all the possible η values, we build the probability-aware index ($\mathcal{I}_{pa}$) that consists of a series of η-indexes for some selected η values. In this way, given query parameters α, β, and η_q, if the η_q-index is built, we can optimally find the (α, β, η_q)-core in the index. Otherwise, we can compute the (α, β, η)-core from the intermediate results fetched from the constructed indexes and avoid searching from scratch. To make such an index practically applicable, we still need to address the following two major challenges.

Challenge 1. *How to select η values for building the η-indexes.* The sizes of the η-indexes do not decrease proportionally as η increases. Thus, if we simply build the η-indexes for some η values with a fixed probability step (e.g., setting η = 0.1, 0.2, ..., 1.0), it is very possible that the adjacent η-indexes are nearly identical, which leads to high redundancy. Thus, the choice of η values will significantly affect the construction of the probability-aware index and its ability to answer queries.

Challenge 2. *Given query α, β, and η_q, how to efficiently retrieve the results if the η_q-index is not built.* If the η_q-index is not built, it is intuitive to find the closest two η values with indexes built ($\eta_1 < \eta_q < \eta_2$). However, the difference between η_1-index and η_2-index can be very large, and it is not easy to combine the information in these two indexes for retrieving the (α, β, η_q)-core.

Select critical η values. To address Challenge 1, in this part, we introduce how to select critical η values. We first provide an overview the probability-aware index.

The probability-aware index ($\mathcal{I}_{pa}$) consists of several η-indexes. Same as the bicore index in deterministic graphs, each η-index has two parts for vertices in $U(\mathcal{G})$ and $L(\mathcal{G})$, denoted by η-$\mathcal{I}_{pa}^U$ and η-$\mathcal{I}_{pa}^L$, respectively. Before presenting the index structure, we first introduce the concept of α-/β-offset.

Definition 6. **α-/β-offset.** *Given a vertex $u \in V(\mathcal{G})$, a probabilistic threshold η, and an α value, its α-offset denoted as $s_a(u, \alpha, \eta)$ is the maximal β value where u can be contained in an (α, β, η)-core. Symmetrically, the β-offset $s_b(u, \beta, \eta)$ of u is the maximal α value where u can be contained in an (α, β, η)-core.*

Each η-index has three levels including two levels of pointer tables and one level of vertex blocks. Specifically, in η-$\mathcal{I}_{pa}^U$, each block is associated with an (α, β) combination, which is pointed by the pointer stored in η-$\mathcal{I}_{pa}^U[\alpha][\beta]$. Each vertex block contains the vertices $u \in U(\mathcal{G})$ with α-offset $= \beta$. η-$\mathcal{I}_{pa}^L$ also has the similar structure with η-$\mathcal{I}_{pa}^U$. We show the probability-aware index in the following example.

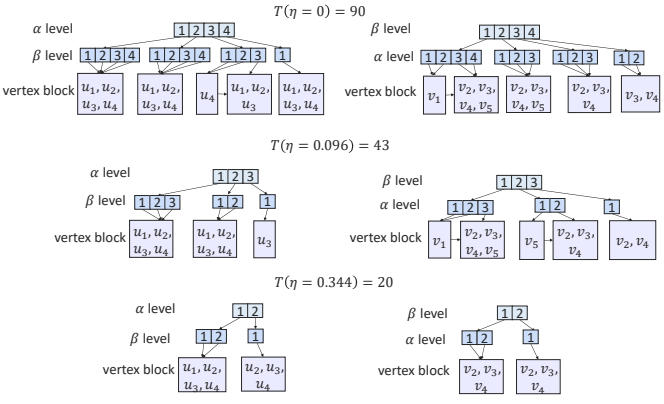


Fig. 3: The probability-aware index $\mathcal{I}_{pa}$ of $\mathcal{G}$

Example 3. *Consider the uncertain bipartite graph $\mathcal{G}$ in Figure 1 and the probability-aware index $\mathcal{I}_{pa}$ in Figure 3. We select three probabilistic thresholds $\eta = 0$, 0.096, and 0.344. Unlike $\mathcal{I}_f$, we build an η-index for each probabilistic threshold instead of storing them in the vertex blocks. Note that the vertex blocks in η-$\mathcal{I}_{pa}^U$ and η-$\mathcal{I}_{pa}^L$ might share the same*

(α, β, η) *value, but their meanings are different. For example, in the* η*-index with* $\eta = 0$*,* u_1 *in the vertex block* $(1, 3)$ *means that* u_1 *is in the* $(1, 3, 0)$-core*. In addition,* v_2 *in the vertex block* $(1, 3)$ *means that* v_2 *is in the* $(3, 1, 0)$-core*.*

To find critical η values, we observe that the number of vertices in the η-index is decreased with the increasing of η. Based on this, although the contents in the η-indexes cannot be foreknown, we can estimate the ability of answering queries using the η-indexes (using $\mathcal{T}(\eta)$) to avoid building η-indexes with similar contents. Here $\mathcal{T}(\eta)$ denote the total number of vertices in the (α, β, η)-cores that can be queried from the η-index. The equation for calculating $\mathcal{T}(\eta)$ is as follows.

$$\mathcal{T}(\eta) = \sum_{u \in U(\mathcal{G})} \sum_{\alpha \in [1, deg(u)]} s_a(u, \alpha, \eta) + \sum_{v \in L(\mathcal{G})} \sum_{\beta \in [1, deg(v)]} s_b(v, \beta, \eta) \tag{6}$$

However, the cost of computing all α-/β-offsets is relatively large. This is equivalent to do an (α, β, η)-core decomposition in the entire graph. The following observation allows us to estimate an upper bound for $\mathcal{T}(\eta)$ without having to compute all α-/β-offsets.

Lemma 2. *Given an uncertain bipartite graph* $\mathcal{G}$*, for an upper vertex* u*,* $s_a(u, \alpha, \eta) \leq max_{v \in N(u)} \eta$*-*$degree(v)$*. For a lower vertex* v*,* $s_b(v, \beta, \eta) \leq max_{u \in N(v)} \eta$*-*$degree(u)$*.*

Proof. This lemma is immediate according to Definition 2 and Definition 6. □

Based on the above lemma, we can use $max_{u \in N(v)} deg_\eta(u)$ to estimate $s_a(u, \alpha, \eta)$ and simplify Equation 6 as follows:

$$\begin{aligned} \mathcal{T}(\eta) \leq T(\eta) &= \sum_{u \in V(\mathcal{G})} deg(u) \cdot max_{v \in N(u)} deg_\eta(v) \\ &= m \cdot \sum_{u \in V(\mathcal{G})} max_{v \in N(u)} deg_\eta(v) \end{aligned} \tag{7}$$

To compute $T(\eta)$ using Equation 7, we first compute the η-degree for each vertex $u \in V(\mathcal{G})$, which takes $O(m \cdot \Delta)$ time in total. Then, for all the vertices, we can get the maximum η-degree of their neighbors in $O(m)$ time. Thus, the total complexity of computing $T(\eta)$ is $O(m \cdot \Delta)$.

Computing $T(\eta)$ allows us to manipulate the size of the η-indexes. In other words, we can increase η until $T(\eta)$ significantly decreases, which indicates the shrink of the size of the η-index. Specifically, we first build the η_1-index with η_1 = 0. Then, we use a binary search approach to find the smallest η_2 value with $T(\eta_2) \leq \frac{1}{2} \cdot T(\eta_2)$. We repeat this process and halve the value of $T(\eta)$ each time to select a new η until the η-index cannot support any (α, β, η)-core query with non-trivial values of α and β ($\alpha > 1$ or $\beta > 1$). Finally we will get $log(T(0))$ number of η-indexes and the space complexity of $\mathcal{I}_{pa}$ is bounded by $O(m \cdot log(m))$ since $T(0)$ is bounded by $O(m)$, and each η-index is bounded by $O(m)$ as proved in [15].

B. Query with different probabilities

To address Challenge 2, in this section, we introduce a local search algorithm to find the (α, β, η)-core based on $\mathcal{I}_{pa}$. Given query parameters α, β, and η_q, if the η_q-index is built, we directly respond to the query by fetching the vertices of the (α, β, η_q)-core from the η_q-index. Otherwise, the querying process begins from the η_1-index and the η_2-index, which has the closest two η values with η_q ($\eta_1 < \eta_q < \eta_2$). We use these two indexes to find the vertices in (α, β, η_1)-core and (α, β, η_2)-core to compute (α, β, η)-core. Before presenting the algorithm, we first introduce the following two vertex sets in the querying process.

- The vertex set S_2 that contains the vertices must belong to (α, β, η_q)-core. According to Lemma 1, (α, β, η_2)-core is a subset of (α, β, η)-core. Thus, we can initialize S_2 as the vertices in (α, β, η_2)-core.
- The vertex set S_1 that contains the vertices may belong to in (α, β, η_q)-core. According to the Lemma 1, (α, β, η)-core is a subset of (α, β, η_1)-core. Thus, we can initialize S_1 as the vertices in (α, β, η_1)-core excluding the vertices in (α, β, η_2)-core (i.e., S_2).

Based on these two sets, the key to computing the (α, β, η_q)-core is to verify the vertices in S_1. The following lemmas can help us achieve this efficiently.

Lemma 3. *Consider a vertex* $u \in S_1$*.* $N_{ub}(u)$ *denotes* u*'s neighbors in* $S_1 \cup S_2$*, and* $d_{ub}(u)$ *denotes the upper bound of* η_q-degree *of* u*, which is computed based on* $N_{ub}(u)$ *using Equation 4. If* $d_{ub}(u)$ *does not satisfy the degree constraint (i.e.,* $d_{ub}(u) < \alpha$ *if* $u \in U(\mathcal{G})$ *or* $d_{ub}(u) < \beta$ *if* $u \in L(\mathcal{G})$*), then* u *cannot belong to the* (α, β, η_q)*-core.*

Proof. According to Definition 3, all the neighbors of u that can belong to the (α, β, η_q)-core must be contained in $S_1 \cup S_2$. Thus, if $d_{ub}(u)$ does not satisfy the degree constraint, it cannot be contained in the (α, β, η_q)-core. □

Lemma 4. *Consider a vertex* $u \in S_1$*.* $N_{lb}(u)$ *denotes* u*'s neighbors in* S_2*, and* $d_{lb}(u)$ *denotes the lower bound of* η_q-degree *of* u*, which is computed based on* $N_{lb}(u)$ *using Equation 4. If* $d_{lb}(u)$ *satisfy the degree constraint, then* u *must belong to the* (α, β, η_q)*-core.*

Proof. This lemma can be proved similarly as Lemma 3. □

Based on the above lemmas, we can iteratively verify and update the vertices in S_1. Note that in each iteration, we only need to check a small portion of vertices that are affected due to the changes in the former iteration. The details of the algorithm are shown in Algorithm 4. We first check whether there exists a η_q-index in the probability-aware index in Line 1. Otherwise, in Lines 3-5 we initialize η_1 and η_2 as the closest two η values with η_q that have η-indexes built in $\mathcal{I}_{pa}$ ($\eta_1 < \eta_q < \eta_2$). We initialize S_1 and S_2 based on the η_1-index and η_2-index. After that, in Line 6, we use S_{ub} and S_{lb} to store the vertices whose upper/lower bounds need to be calculated. For the main loop in Lines 7-24, we verify the vertices in S_{ub} and S_{lb}. Specifically, we calculate the upper bound of η_q-degree of $u \in S_{ub}$ in Line 11. According to Lemma 3, if $d_{ub}(u)$ violates the degree constraint, we remove u from S_1 in Lines

Algorithm 4: Query based on $\mathcal{I}_{pa}$

Input: An uncertain graph $\mathcal{G} = (V = (U, L), E, p)$, two integers α, β, a probabilistic threshold η_q, and $\mathcal{I}_{pa}$
Output: the vertices of (α, β, η)-core in $\mathcal{G}$

1 **if** η_q*-index is built in* $\mathcal{I}_{pa}$ **then**
2 | **return** (α, β, η)-core based on the η_q-index;
3 $\eta_1, \eta_2 \leftarrow$ the closest two η values with η_q that have η-indexes built in $\mathcal{I}_{pa}$ $(\eta_1 < \eta_q < \eta_2)$;
4 $S_2 \leftarrow V((\alpha, \beta, \eta_2)$-core) based on η_2-index;
5 $S_1 \leftarrow V((\alpha, \beta, \eta_1)$-core$)\backslash S_2$ based on η_1-index;
6 $S_{ub} \leftarrow S_1$; $S_{lb} \leftarrow S_1$;
7 **while** $S_{ub} \cup S_{lb} \neq \emptyset$ **do**
8 | $S'_{ub} \leftarrow \emptyset$; $S'_{lb} \leftarrow \emptyset$;
9 | **foreach** $u \in S_{ub}$ **do**
10 | | $N_{ub}(u) \leftarrow N(u) \cap (S_1 \cup S_2)$;
11 | | $d_{ub}(u) \leftarrow$ the upper bound of η_q-degree of u w.r.t. $N_{ub}(u)$;
12 | | **if** $d_{ub}(u)$ *does not satisfy the degree constraint* **then**
13 | | | S_1.remove(u);
14 | | | **foreach** $v \in (N(u) \cap S_1)$ **do**
15 | | | | S'_{ub}.add(v);
16 | **foreach** $u \in S_{lb}$ **do**
17 | | $N_{lb}(u) \leftarrow N(u) \cap S_2$;
18 | | $d_{lb}(u) \leftarrow$ the lower bound of η_q-degree of u w.r.t. $N_{lb}(u)$;
19 | | **if** $d_{lb}(u)$ *satisfies the degree constraint* **then**
20 | | | move u from S_1 to S_2;
21 | | | **foreach** $v \in (N(u) \cap S_1)$ **do**
22 | | | | S'_{lb}.add(v);
23 | $S_{ub} \leftarrow S'_{ub}$; $S_{lb} \leftarrow S'_{lb}$;
24 $S_2 \leftarrow S_1 \cup S_2$;
25 **return** S_2

12-13. For each vertex v in $N(u) \cap S_1$, $d_{ub}(v)$ can be affected due to the removal of u and we add v in S'_{ub} in Line 15, and update the $d_{ub}(v)$ in the next round. After that, we calculate the lower bound of η_q-degree of $u \in S_{lb}$ in Line 18. Based on Lemma 4, if $d_{lb}(u)$ satisfies the degree constraint, we move u from S_1 to S_2 in Lines 19-21. Then, for each vertex v in $N(u) \cap S_1$, we add v in S'_{lb} in Line 23, and update $d_{lb}(v)$ in the next iteration. We repeat this process until $S_{ub} \cup S_{lb} = \emptyset$. Note that at the end of the while loop, S_1 may still contains several vertices. We find that the upper bound of the η-degree of the remaining vertices always meets the degree constraint. Thus, we can directly put them into S_2. Finally, we return S_2 as the result.

Complexity analysis. The time complexity of Algorithm 4 is $O(\sum_{u \in S_1} Deg(u)^2 \cdot deg_{\eta_1}(u))$, where $S_1 = V((\alpha, \beta, \eta_1)$-core$)\backslash V((\alpha, \beta, \eta_2)$-core$)$. For each $u \in S_1$, we need to compute its η-degree in $O(Deg(u) \cdot deg_{\eta}(u))$ time. In addition, each vertex $u \in S_1$ is removed at most once and there are at most $O(Deg(u))$ vertices affected. Thus, the time complexity of Algorithm 4 is $O(\sum_{u \in S_1} Deg(u)^2 \cdot deg_{\eta_1}(u))$.

Example 4. *Consider the uncertain bipartite graph $\mathcal{G}$ in Figure 1 and the probability-aware index $\mathcal{I}_{pa}$ of $\mathcal{G}$ in Figure 3. Given query parameters $\alpha = 2$, $\beta = 1$, and $\eta_q = 0.3$. Since* 0.3-index *is not built in the $\mathcal{I}_{pa}$, Algorithm 4 finds the closest two η values with indexes built (i.e., $\eta_1 = 0.096$ and $\eta_2 = 0.344$). Then, based on the η_1-index and the η_2-index, we get $S_1 = \{u_1, v_1, v_5\}$, and $S_2 = \{u_2, u_3, u_4, v_2, v_3, v_4\}$. After that, we need to calculate the upper/lower bounds of the η_q-degrees of the three vertices in S_1. We first compute the upper bounds. By considering the union of u_1's neighbors in S_1 and S_2, $N_{ub}(u_1) = \{v_1, v_2, v_3, v_4\}$ and $d_{ub}(u_1) = 2$ w.r.t $N_{ub}(u_1)$. Since $d_{ub}(u_1)$ satisfies the degree constraint (i.e., $d_{ub}(u_1) \geq 2$), u_1 is not removed from S_1 in this iteration. We can also get $d_{ub}(v_1) = d_{ub}(v_5) = 0 < 1$. Thus, these two vertices are removed from S_1, and we add u_1 into S'_{ub} since it is a neighbor of v_1 in S_1. After that, we compute the lower bounds. To compute $d_{lb}(u_1)$, the algorithm finds u_1's neighbors in S_2 (i.e., $N_{lb}(u_1) = \{v_2, v_3, v_4\}$). Then, we can compute $d_{lb}(u_1) = 1$. Since $d_{lb}(u_1) < 2$, the algorithm does not move u_1 from S_1 to S_2. In the second iteration, S_1 only contains u_1 and $d_{ub}(u_1) = 1 < 2$. Thus, we remove u_1 from S_1. Finally, we return the vertices $\{u_2, u_3, u_4, v_2, v_3, v_4\}$ in S_2 as the vertices in $(2, 1, 0.3)$-core.*

C. Index construction

In this subsection, we present the algorithms for building the probability-aware index $\mathcal{I}_{pa}$.

A bottom-up approach. Since each η-index is actually a bicore index under the η threshold, a straightforward way to build the η-indexes is to adopt the algorithm in [15]. The details of the algorithm $\mathsf{IC_{pa}}$-BU is shown in Algorithm 5. We start from $\eta_1 = 1$. To build the η_1-index, we call the **Bottom-up Sub-index Construction** function (Line 3). We set δ_η as the maximum value s.t. $(\delta_{\eta_1}, \delta_{\eta_1}, \eta_1)$-core $\neq \emptyset$. Then, for $\alpha \in [1, \delta_{\eta_1}]$, we first obtain the $(\alpha, 1, \eta)$-core using the online computation algorithm (Line 15). Then, we follow a peeling paradigm to compute the α-offsets of the upper vertices and update some β-offsets of the lower vertices. In each iteration, we remove the vertices s.t. their η_1-degree does not satisfy the degree constraint. Then, we run β from 1 to δ_η to compute the remaining α/β-offsets in a similar way as Lines 15 - 34. After that, we organize the α/β-offsets into the η_1-index and combine the η_1-index into $\mathcal{I}_{pa}$. To build the second sub-index (the η_2-index), we need to compute $T(\eta_1)$ based on Equation 7 at first. Then, η_2 is the smallest η value s.t. $T(\eta_j) \leq \frac{1}{2} \cdot T(\eta_{j-1})$ and it can be easily found using the binary search. Note that when finding the η_2, we need to set a precision threshold γ (e.g., 0.001) to avoid exhausting search. The other η-indexes can also be built in a similar fashion.

Complexity analysis. The time complexity of Algorithm 5 is bounded by $O(log(m) \cdot \delta \cdot \Delta \cdot m)$. Since $T(0)$ is bounded by $O(m)$, there are $O(log(m))$ sub-indexes to be built. Note that the time complexity for building the bicore index is $O(\delta \cdot m)$ [15] since there are δ iterations. Building each η-index needs to compute the η-degree (which needs $O(\Delta \cdot m)$ time in total for each iteration, and Δ represents the maximum η-degree in each iteration) rather than the vertex degree, the time complexity of building the η-index is $O(\delta \cdot \Delta \cdot m)$. In total, the time complexity of Algorithm 5 is bounded by $O(log(m) \cdot \delta \cdot \Delta \cdot m)$.

Example 5. *Consider the probability-aware index $\mathcal{I}_{pa}$ in Figure 3. If the* $\mathsf{IC_{pa}}$-BU *algorithm is used to build $\mathcal{I}_{pa}$, we first initialize η_1 to* 0 *and build the η_1-index which is similar as the bicore-index in [15] since the probability threshold is zero. We compute $T(0) = 90$ and get $\eta_2 = 0.096$ since $T(0.096) = 43 \leq \frac{1}{2} \cdot T(0)$. Then, given $\eta_2 = 0.096$, we build*

Algorithm 5: $\mathsf{IC_{pa}}$-BU

Input: An uncertain graph $\mathcal{G} = (V = (U, L), E, p)$
Output: $\mathcal{I}_{pa}$

1 $\eta_1 \leftarrow 0$; $j \leftarrow 1$;
2 **while** $\eta_j < 1$ **do**
3 $\quad \eta_j$-index $\leftarrow$ **Bottom-up Sub-index Construction**(η_j, $\mathcal{G}$);
4 $\quad$ **if** *any* (α, β, η_j)*-core with* $\alpha > 1$ *or* $\beta > 1$ *cannot be queried from the* η_j*-index* **then**
5 $\quad\quad$ **break**;
6 $\quad \mathcal{I}_{pa} \leftarrow \mathcal{I}_{pa} \cup \eta_j$-index;
7 $\quad$ compute $T(\eta_j)$ based on Equation 7;
8 $\quad j \leftarrow j + 1$;
9 $\quad \eta_j \leftarrow$ the smallest η value s.t. $T(\eta_j) \leq \frac{1}{2} \cdot T(\eta_{j-1})$;
10 **return** $\mathcal{I}_{pa}$
11 **Function Bottom-up Sub-index Construction**(η, $\mathcal{G}$)
12 initialize all the α/β-offsets as zero;
13 $\delta_\eta \leftarrow$ the maximum value such that $(\delta_\eta, \delta_\eta, \eta)$-core $\neq \emptyset$;
14 **for** $\alpha = 1$ **to** δ_η **do**
15 $\quad \mathcal{G}' \leftarrow$ the $(\alpha, 1, \eta)$-core of $\mathcal{G}$;
16 $\quad$ compute $deg_\eta(u)$ for all $u \in V(\mathcal{G}')$;
17 $\quad$ **while** $\exists u \in U(\mathcal{G}')$ *s.t.* $deg_\eta(u) < \alpha$ **do**
18 $\quad\quad$ **foreach** $v \in N(u)$ **do**
19 $\quad\quad\quad$ update $deg_\eta(v)$;
20 $\quad\quad$ remove u from $\mathcal{G}'$;
21 $\quad$ **while** $\mathcal{G}' \neq \emptyset$ **do**
22 $\quad\quad \beta \leftarrow min_{v \in L(\mathcal{G}')} deg_\eta(v)$;
23 $\quad\quad$ **while** $\exists v \in L(\mathcal{G}')$ *s.t.* $\deg_\eta(v) \leq \beta$ **do**
24 $\quad\quad\quad$ **foreach** $u \in N(v)$ **do**
25 $\quad\quad\quad\quad$ update $deg_\eta(u)$;
26 $\quad\quad\quad$ **for** $i = 1$ **to** β **do**
27 $\quad\quad\quad\quad$ **if** $s_b(v, i, \eta) < \alpha$ **then**
28 $\quad\quad\quad\quad\quad s_b(v, i, \eta) \leftarrow \alpha$;
29 $\quad\quad\quad$ **while** $\exists u \in U(\mathcal{G}')$ *s.t.* $\deg_\eta(u) < \alpha$ **do**
30 $\quad\quad\quad\quad s_a(u, \alpha, \eta) \leftarrow \beta$;
31 $\quad\quad\quad\quad$ **foreach** $w \in N(u)$ **do**
32 $\quad\quad\quad\quad\quad$ update $deg_\eta(w)$;
33 $\quad\quad\quad\quad$ remove u from $\mathcal{G}'$;
34 $\quad\quad\quad$ remove v from $\mathcal{G}'$;
35 **for** $\beta = 1$ **to** δ_η **do**
36 $\quad \mathcal{G}' \leftarrow$ the $(1, \beta, \eta)$-core of $\mathcal{G}$;
37 $\quad$ run Lines 14 - 32 by interchanging u with v, U with L, α with β;
38 organize α-offset of each vertex $u \in U(\mathcal{G})$ and β-offset of each vertex $v \in L(\mathcal{G})$ into the η-index;
39 **return** η*-index*

the η_2*-index in a bottom-up manner. We get* $\eta_3 = 0.344$ *since* $T(0.344) = 20 \leq \frac{1}{2} \cdot T(0.096)$*, and the* η_3*-index can be built similarly. At last, since the* η*-index with larger* η *value cannot support any* (α, β, η)*-core query with non-trivial values of* α *and* β *(*$\alpha > 1$ *or* $\beta > 1$*), we terminate the* $\mathsf{IC_{pa}}$-BU *algorithm.*

A top-down approach. In $\mathsf{IC_{pa}}$-BU, we follow a bottom-up manner to compute the α/β-offsets for each sub-index. In this process, we fix α (or β) and compute the (α, β, η)-core from the smallest β (or α) value. In this manner, the vertices with high η-degrees can be updated many times. Since the update of η-degree is expensive, this compromises the efficiency of index construction. To address this issue, we present a top-down index construction algorithm that prioritizes processing the (α, β, η)-cores with large degree constraints, which contain many vertices with high η-degrees. In this manner, the computation of these vertices with high η-degrees can be limited in a smaller subgraph compared with the original graph. In addition, these vertices with high η-degrees in the (α, β, η)-cores with large α or β will occur in the (α, β, η)-cores with smaller α or β as well. Thus, the η-degrees of these vertices do not have to be updated as we visit the (α, β, η)-cores with smaller degree constraints. In this manner, the total number of η-degree updates can be significantly decreased compared to the bottom-up approach. Before presenting the algorithm, we give the following lemma.

Lemma 5. *Given* α*,* β*, and* η*,* (α, β, η)*-core* $\subseteq (\alpha, \beta)$*-core.*

Proof. This lemma is immediate according to Definition 1 and Definition 3. □

Based on the above lemma, when finding the (α, β, η)-core, we can first obtain the (α, β)-core to limit the search space. The details of the top-down index construction algorithm $\mathsf{IC_{pa}}$-TD is shown in Algorithm 6. The main process of $\mathsf{IC_{pa}}$-TD is the same as $\mathsf{IC_{pa}}$-BU and the only difference is that we use the function **Top-down Sub-index Construction** to compute each sub-index. When running the **Top-down Sub-index Construction**, we first initialize all the α/β-offsets as zero. Then, for $\alpha \in [1, \delta_\eta]$, we set β from $\beta_{max}(\alpha)$ to 1 to find the (α, β, η)-cores in a top-down manner. Here $\beta_{max}(\alpha)$ is the maximal β value such that the (α, β)-core exists. In each iteration, we first obtain the (α, β)-core from the deterministic graph of $\mathcal{G}$ since (α, β, η)-core $\subseteq (\alpha, \beta)$-core according to Lemma 5. Note that for each α, all the non-empty (α, β)-cores ($\beta \in [1, \beta_{max}(\alpha)]$) can be pre-computed in $O(m)$ time by following the peeling paradigm [14]. Then, we put the vertices and edges of (α, β)-core into $\mathcal{G}'$ and compute $deg_\eta(u)$ for each $u \in V(\mathcal{G}')$ incrementally (Lines 8-9). After that, we use the similar peeling approach as $\mathsf{IC_{pa}}$-BU to find the (α, β, η)-core (i.e., the α/β-offsets of the vertices). Note that when removing a lower vertex v that does not satisfy the degree constraint, we only need to update the η-degree of its neighbor u with $s_a(u, \alpha, \eta) < \beta$. This rule ensures that we do not need to update the η-degree of the vertices in the (α, β', η)-core (with $\beta' > \beta$) when computing the (α, β, η)-core. Note that it will also be applied when removing an upper vertex. After obtaining the (α, β, η)-core, we assign the α/β-offsets to the vertices in it (Lines 25-31). After the computation process from Lines 5-32, we organize the α/β-offsets into the η-index.

Complexity analysis. Since $T(0)$ is bounded by $O(m)$, there are $O(log(m))$ sub-indexes to be built. The time complexity of building each sub-index is $O(\sum_{\alpha \in [1,\delta], \beta \in [1, \beta_{max}(\alpha)]} (|(\alpha, \beta)\text{-core}| - |F|) \cdot \Delta_{\alpha,\beta})$. Here, $|(\alpha, \beta)\text{-core}|$ denotes the size of the (α, β)-core, $|F|$ denotes the number of vertices that do not need η-degree updating in each iteration, and $\Delta_{\alpha,\beta}$ denotes the maximum η-degree in the (α, β)-core. In total, the time complexity of Algorithm 6 is bounded by $O(log(m) \cdot \sum_{\alpha \in [1,\delta], \beta \in [1, \beta_{max}(\alpha)]} (|(\alpha, \beta)\text{-core}| - |F|) \cdot \Delta)$.

Example 6. *Here we show how to construct the* η*-index with* $\eta = 0.096$ *using the* ***Top-down Sub-index Construction*** *function. We show the process when* $\alpha = 1$ *and begin with* $\beta = 4$*. Since there is no* $(1, 4, \eta)$-core *in* $\mathcal{G}$*, we decrease* β *by 1. When* $\beta = 3$*, we let* $\mathcal{G}'$ *represent the* $(1, 3)$*-core*

Algorithm 6: $\mathsf{IC_{pa}}$-TD

Input: An uncertain graph $\mathcal{G} = (V = (U, L), E, p)$
Output: $\mathcal{I}_{pa}$

run Lines 1-10 of Algorithm 5, use **Top-down Sub-index Construction** in Line 3;
Function Top-down Sub-index Construction(η, $\mathcal{G}$)
initialize all the α/β-offsets as zero;
$\delta_\eta \leftarrow$ the maximum value such that $(\delta_\eta, \delta_\eta, \eta) \neq \emptyset$;
for $\alpha = 1$ **to** δ_η **do**
  $\beta = \beta_{max}(\alpha)$;
  while $\beta > 0$ **do**
    $\mathcal{G}' \leftarrow$ the (α, β)-core of $\mathcal{G}$;
    compute $deg_\eta(u)$ for each $u \in V(\mathcal{G}')$;
    while $\exists u \in U(\mathcal{G}')$ *s.t.* $deg_\eta(u) < \alpha$ **do**
      for $v \in N(u)$ **do**
        if $s_b(v, \beta, \eta) < \alpha$ **then**
          update $deg_\eta(v)$;
      remove u from $\mathcal{G}'$;
    while $\exists v \in L(\mathcal{G}')$ *s.t.* $\deg_\eta(v) < \beta$ **do**
      foreach $u \in N(v)$ **do**
        if $s_a(u, \alpha, \eta) < \beta$ **then**
          update $deg_\eta(u)$;
      while $\exists u \in U(\mathcal{G}')$ *s.t.* $\deg_\eta(u) < \alpha$ **do**
        foreach $w \in N(u)$ **do**
          if $s_b(w, \beta, \eta) < \alpha$ **then**
            update $deg_\eta(w)$;
        remove u from $\mathcal{G}'$;
      remove v from $\mathcal{G}'$;
    foreach *vertex* x *in* $\mathcal{G}'$ **do**
      if $x \in U(\mathcal{G}')$ *with unassigned* α*-offset* **then**
        $s_a(x, \alpha, \eta) \leftarrow \beta$;
      if $x \in L(\mathcal{G}')$ **then**
        for $i = 1$ **to** β **do**
          if $s_b(x, i, \eta) < \alpha$ **then**
            $s_b(x, i, \eta) \leftarrow \alpha$;
    $\beta \leftarrow \beta - 1$;
for $\beta = 1$ **to** δ_η **do**
  run Lines 6-32 by interchanging u with v, U with L, α with β;
organize α-offset of each vertex $u \in U(\mathcal{G})$ and β-offset of each vertex $v \in L(\mathcal{G})$ into the η-index;
return η*-index*

containing $\{u_1, u_2, u_3, u_4, v_2, v_3, v_4\}$*. We iteratively remove the vertices in* $\mathcal{G}'$ *until we get the* $(1, 3, \eta)$*-core containing* $\{u_1, u_2, u_3, u_4, v_2, v_4\}$*. Then, we set* $s_a(u_1, 1, \eta)$*,* $s_a(u_2, 1, \eta)$*,* $s_a(u_3, 1, \eta)$*,* $s_a(u_4, 1, \eta)$ *to 3. We also set* $s_b(v_2, i, \eta)$ *and* $s_b(v_4, i, \eta)$ *to 1 (*$i \in [1, 3]$*). After that, when* $\beta = 2$*, we compute the* $(1, 2, \eta)$*-core from the* $(1, 2)$*-core and set* $s_b(v_3, i, \eta)$ *and* $s_b(v_5, i, \eta)$ *to 1 (*$i \in [1, 2]$*). Note that when removing the vertices that violate the degree constraints, we do not need to update the* η*-degree of its neighbors in the* $(1, 3, \eta)$*-core. For instance, when removing* v_1 *(*$deg_\eta(v_1) = 1 < \beta = 2$*), we can skip updating the* η*-degree of* u_1 *since* u_1 *belongs to the* $(1, 3, \eta)$*-core. Finally, we compute the* $(1, 1, \eta)$*-core and assign* $s_b(v_1, 1, \eta) = 1$.

V. Experimental evaluations

In this section, we evaluate the uncertain (α, β, η)-core model and the proposed algorithms through extensive experiments. First, we validate the effectiveness of the (α, β, η)-core model via a case study. Then, we evaluate the efficiency and scalability of the proposed algorithms with different parameter settings and datasets. *We terminate an experiment process if its running time exceeds* 10^5 *seconds.*

A. Experiments setting

Algorithms. Our empirical studies are conducted against the following designs: We evaluate the query performance of the following algorithms: 1) the online query algorithm $\mathsf{Q_o}$; 2) the full index-based query algorithm $\mathsf{Q_f}$; and 3) the probability-aware index-based query algorithm $\mathsf{Q_{pa}}$. We also analyze the index construction time and sizes of the full index $\mathcal{I}_f$ and the probability-aware index $\mathcal{I}_{pa}$. All algorithms are implemented in C++, and all experiments are tested on a Linux server with Intel Xeon CPU E3-1231 3.4 GHz and 16GB main memory.

TABLE II: Summary of Datasets

Dataset	$\|E\|$	$\|U\|$	$\|L\|$	d_{max}	δ
Unicode-Lang (UL)	1.26K	0.87K	0.25K	141	4
Producers (PR)	207.27K	187.68K	48.83K	512	6
Youtube (YT)	293K	94K	30K	7592	20
Github (GH)	440K	56K	120K	3676	39
BookCrossing (BX)	1.15M	445.8K	105.3K	13,601	41
Stack-Overflow (SO)	1.30M	545.2K	96.7K	6,119	22
Teams (TM)	1.37M	935.6K	901.2K	2,671	9
Actor-Movies (AM)	1.47M	127K	383K	647	14
Wiki-en (WC)	3.80M	2.04M	1.85M	11,593	18
DBLP (DB)	12.3M	1.95M	5.62M	1386	48

Datasets. We use 10 real datasets in our experiments, which are `Unicode-Lang` (UL), `Producers` (PR), `Github` (GH), `BookCrossing` (BX), `StackOverflow` (SO), `Teams` (TM), `Actor-Moives` (AM), `Wiki-en` (WC), and `DBLP` (DB). These datasets can be downloaded from the website KONECT (http://konect.cc/). Note that for each dataset, the probabilities of edges are assigned to follow the exponential distribution by default, which has been widely used in uncertain graph literature [6], [7], [27], [28]. Specifically, for a given bipartite graph, we assign a random weight (chosen from [1, 10]) to each edge and use the exponential distribution with the expectation 2 to generate the probabilities of edges.

Table II includes the statistics of the 10 datasets. $|E|$ is the number of edges in the graph. $|U|$ and $|L|$ are the number of vertices in the upper and lower layers of the graph. d_{max} is the maximum degree in the graph, and δ is the maximum integer such that the (δ, δ)-core exists in the graph. M denotes 10^6 and K denotes 10^3.

B. Effectiveness evaluation

TABLE III: Statistics of query results, $\alpha = 50$ and $\beta = 100$

η	$\|E(C)\|$	$\|U(C)\|$	$\|L(C)\|$	$deg_{avg}(U)$	$deg_{avg}(L)$	p_{avg}
0	562,781	721	1093	86.21	164.29	0.211
0.3	245,732	287	564	61.67	135.46	0.278
0.5	209,760	168	437	76.28	116.73	0.295
0.7	136,233	152	371	56.17	109.37	0.379
0.9	79,314	137	262	52.31	103.21	0.437

In this subsection, we conduct a case study using the DBLP dataset (*https://dblp.org/xml/*). We extract a dataset containing researchers, the paper written by the researchers, and the name of venue where each paper is published and obtain an uncertain bipartite graph formed by researchers (U) and venues (L). To generate the edge probabilities, we first assign the weight of each edge as the total number of

papers published by a researcher in a venue. Then, we use an exponential distribution with expectation 5 to the edge weight to generate the probabilities of edges. We choose the conference and journal papers since 2015, and we can get a bipartite graph with $|U| = 1{,}708{,}390$, $|L| = 7{,}746$, and $|E| = 5{,}664{,}157$.

To form a team with strong expertise to perform the Electrical and Computer Engineering research, we query the (α, β, η)-core by choosing the famous researcher H.Vincent Poor in this field as the query seed. Table III shows the statistics of query results. $|E(C)|$, $|U(C)|$ and $|L(C)|$ denote the total number of edges, researchers and venues in the resulting subgraph, respectively. $deg_{avg}(U)$ and $deg_{avg}(L)$ denote the average η-degree on the researcher side and venue side. p_{avg} denotes the average probability of all the edges. We can observe that with the increasing of the query probability threshold η, (1) the size of the subgraph becomes smaller, (2) the average probability of edges in the subgraph increases, and (3) the average degree of researchers and venues decreases. This case study validates that by considering edge probabilities, we can form a research team with strong expertise using the (α, β, η)-core model.

C. Evaluation of query performance

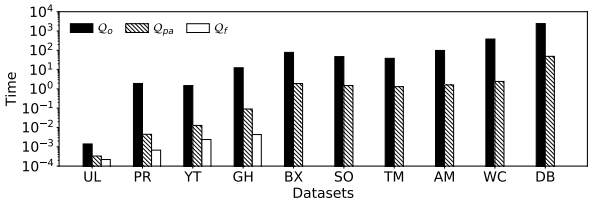
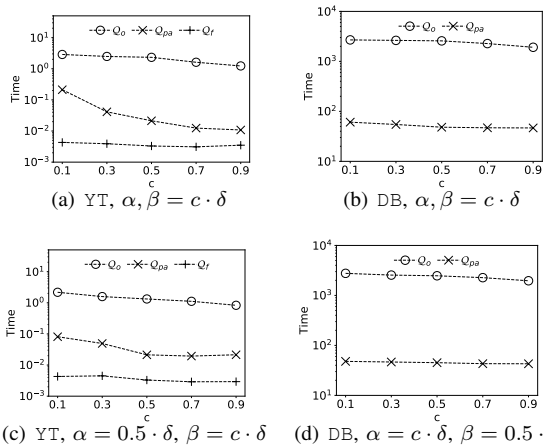


Fig. 4: Retrieving the (α, β, η)-core

In this part, we evaluate the efficiency of the online peeling algorithm ($\mathsf{Q_o}$) and index-based query algorithms ($\mathsf{Q_f}$ and $\mathsf{Q_{pa}}$). Firstly, we evaluate the query time of these algorithms on 10 datasets with default parameters. By default, we set α to $0.4 \cdot \delta$ and β to $0.6 \cdot \delta$, and $\eta = 0.4$. Then we evaluate the effect of parameters α, β and η on the query performance.

Evaluating the query time on all the datasets. In Figure 4, we evaluate the performance of $\mathsf{Q_o}$, $\mathsf{Q_{pa}}$, and $\mathsf{Q_f}$ on all the datasets with default parameters. As expected, the index-based algorithms $\mathsf{Q_{pa}}$ and $\mathsf{Q_f}$ significantly outperform $\mathsf{Q_o}$ on all the datasets. We can observe that $\mathsf{Q_f}$ is faster than $\mathsf{Q_o}$ and $\mathsf{Q_{pa}}$ on small datasets UN, PR, YT, and GH. This is because $\mathsf{Q_f}$ is an algorithm using the full index that stores all the $thres_{(\alpha,\beta)}(u)$ of each vertex with different degree conditions and we only need to identify whether a vertex satisfies the probabilistic threshold η. However, for the datasets with more than 1 million edges, the construction time of the full index $\mathcal{I}_f$ exceeds 10^5 seconds and only $\mathsf{Q_o}$ and $\mathsf{Q_{pa}}$ can retrieve the results for these datasets. Note that $\mathsf{Q_{pa}}$ is one to two orders of magnitude faster than $\mathsf{Q_o}$ on all the datasets. The experimental results show that our probability-aware index $\mathcal{I}_{pa}$ achieves a better trade-off between query efficiency and index construction cost.

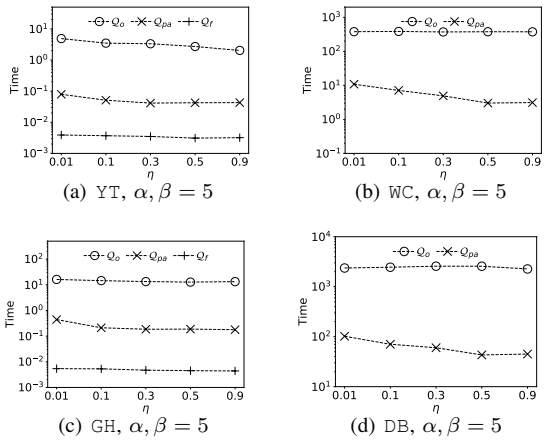
Evaluating the effect of query parameter α and β. We also investigate the query performance by varying parameter α, β and fixing $\eta = 0.4$. In Figure 5(a) and (b), we vary α and β from $0.1 \cdot \delta$ to $0.9 \cdot \delta$ simultaneously. We can observe that $\mathsf{Q_{pa}}$ is one to two orders of magnitude faster than $\mathsf{Q_o}$. In Figure 5(c) and (d), we fix one of α and β at $0.5 \cdot \delta$ and vary the other one from $0.1 \cdot \delta$ to $0.9 \cdot \delta$. The index based querying algorithms are noticeably faster than the online computation algorithm. The $\mathsf{Q_f}$ outperforms $\mathsf{Q_{pa}}$ since it relies on a total index of (α, β, η)-core results.



(a) YT, $\alpha, \beta = c \cdot \delta$ (b) DB, $\alpha, \beta = c \cdot \delta$

(c) YT, $\alpha = 0.5 \cdot \delta$, $\beta = c \cdot \delta$ (d) DB, $\alpha = c \cdot \delta$, $\beta = 0.5 \cdot \delta$

Fig. 5: Retrieving the (α, β, η)-core, varying α and β



(a) YT, $\alpha, \beta = 5$ (b) WC, $\alpha, \beta = 5$

(c) GH, $\alpha, \beta = 5$ (d) DB, $\alpha, \beta = 5$

Fig. 6: Retrieving the (α, β, η)-core, varying η

Evaluating the effect of query parameter η. We further investigate the effect of probabilistic threshold η. We vary η from 0.01 to 0.9 and fix $\alpha, \beta = 5$. In Figure 6, we can observe that the running time of $\mathsf{Q_o}$ and $\mathsf{Q_f}$ remain relatively stable with the change of η. As show in Figure 6(b) and 6(d), since the full index cannot be build on WC and DB within the time limit, the performance of $\mathsf{Q_f}$ is omitted. $\mathsf{Q_{pa}}$ outperforms $\mathsf{Q_o}$ by at least one order of magnitude as expected.

D. Evaluation of indexing techniques

In this section, we evaluate the full index $\mathcal{I}_f$ and the probability-aware index $\mathcal{I}_{pa}$.

Evaluate the index construction time and the index size. *1) Index construction time.* In Figure 7, we can observe that not all indexes can be established within prescribed time in $\mathsf{IC_f}$ and $\mathsf{IC_{pa}}$-BU. Specifically, for $\mathsf{IC_f}$ algorithm, the datasets which are larger than GH cannot be established within 10^5 seconds. For $\mathsf{IC_{pa}}$-BU, the construction time of the datasets

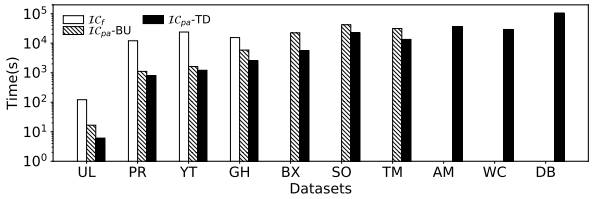


Fig. 7: Comparing index construction time

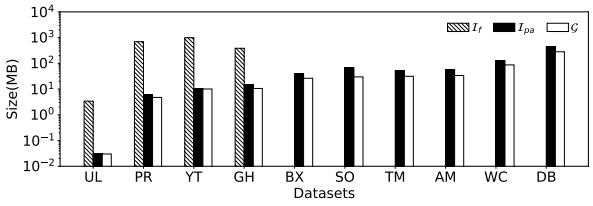
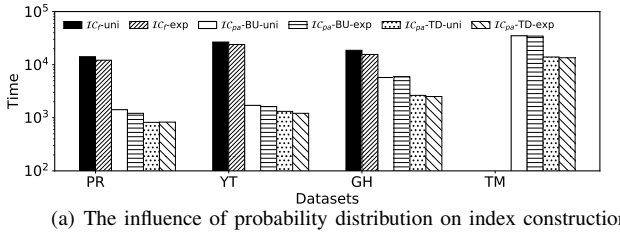


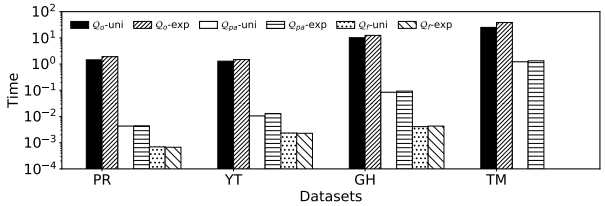
Fig. 8: Comparing index size

with a larger size than TM is also time out. By contrast, with several carefully-designed optimizations, $\mathsf{IC_{pa}}$-TD outperforms $\mathsf{IC_{pa}}$-BU and $\mathsf{IC_f}$, and the index of all the datasets can be efficiently constructed by $\mathsf{IC_{pa}}$-TD.

2) Index size. In Figure 8, we evaluate the index size of $\mathcal{I}_f$ and $\mathcal{I}_{pa}$. Since the full index contains the η-threshold of all the vertices in each (α, β)-core, it's space is larger than probability-aware index by one to two orders of magnitude. Especially, the size of $\mathcal{I}_f$ for the YT graph is close to 1GB while the size of $\mathcal{I}_{pa}$ in YT is only 10.33MB. In addition, the size of $\mathcal{I}_{pa}$ is only $1.03\times$-$2.22\times$ to the graph size, which is very space-compact in practice.



(a) The influence of probability distribution on index construction



(b) The influence of probability distribution on querying process

Fig. 9: The effect of probability distribution

Evaluate the effect of probability distribution. To study the effect of probability distribution of edges, apart from the exponential distribution, we also generate the probabilities of edges under the $[0, 1]$ uniform distribution as done in [7]. As shown in Fig. 9, the algorithms with the suffix "uni" ("exp") are applied on uncertain bipartite graphs with the edge probabilities following the uniform (exponential) distribution. For the index construction, $\mathsf{IC_f}$, $\mathsf{IC_{pa}}$-BU, and $\mathsf{IC_{pa}}$-TD have similar performance under the uniform distribution and the exponential distribution. For the querying process, by default, we set α to $0.4 \cdot \delta$ and β to $0.6 \cdot \delta$, and $\eta = 0.4$. As shown in Fig. 9(b), different probability distributions do not have much impact on the performance of $\mathsf{Q_o}$, $\mathsf{Q_{pa}}$, or $\mathsf{Q_f}$.

VI. Related Work

Here we review the related works of cohesive subgraph models on uncertain graphs and bipartite graphs.

Cohesive subgraph models on uncertain graphs Several cohesive subgraph models are studied on uncertain graphs. To extend k-core to uncertain graphs, Bonchi et al. [8] propose (k, η)-core that uses η-degree to model vertex engagement in uncertain graphs. Our paper adopts η-degree to measure vertex engagement in uncertain bipartite graphs. Yang et al. [6] design an index-based optimal algorithm for retrieving (k, η)-core in uncertain graphs. Very recently, Dai et al. [7] propose an algorithm to compute (k, η)-core that can reduce inaccuracies from floating-point number divisions. In [29], a multi-stage graph peeling algorithm is designed that focuses on mining the dense subgraphs captured by (k, η)-core. Apart from (k, η)-core, Peng et al. [30] propose (k, θ)-core that requires the probability of a vertex appearing in the k-core exceeds θ. Truss-based models are also proposed on uncertain graphs [9], [10], [31]. Zou et al. [10] propose an algorithm to find highly probable k-trusses of an uncertain graph. Huang et al. [9] use dynamic programming to decompose an uncertain graph into maximal k-trusses. Esfahani et al. define h-index of edges to estimate the truss-values progressively. Mining maximal cliques on uncertain graphs is also studied [12], [13].

Cohesive subgraph models on bipartite graphs. Various cohesive subgraph models are proposed on bipartite graphs. (α, β)-core [14], [15] is a representative model extended from k-core, which imposes different degree constraints on vertices of different layers. Specifically, a space-compact index is proposed in [15], which stores the upper and lower vertices of all (α, β)-cores separately. Variants of (α, β)-core are also proposed in the literature [21], [32], [33]. Other models rely on the butterfly [34] (i.e., (2x2)-biclique) structure to ensure the structural cohesiveness of the subgraph, such as k-bitruss [17], k-wing [35], [36], k-tip [35], and τ-strengthened (α, β)-core [32]. In addition, different variants of biclique (a complete bipartite graph) are among the densest and the most cohesive subgraph models, including maximum edge biclique [18], maximal biclique numeration [37], [38], and maximum balanced biclique [39].

VII. Conclusion

In this paper, we propose a novel model (α, β, η)-core, which is the first cohesive subgraph model on uncertain bipartite graphs. To support efficient queries of (α, β, η)-core, we design a basic full index that can fetch the vertices of arbitrary (α, β, η)-core in optimal time complexity. In order to strike a balance between index space cost and query processing time, we propose a probability-aware index with bounded space complexity. We also propose efficient algorithms to construct this index in both bottom-up and top-down manner. Extensive experiments validate the effectiveness of the (α, β, η)-core model and the efficiency of the proposed algorithms.

REFERENCES

[1] C. M. O'Connor, J. U. Adams, and J. Fairman, "Essentials of cell biology," *Cambridge, MA: NPG Education*, vol. 1, p. 54, 2010.

[2] G. A. Pavlopoulos, P. I. Kontou, A. Pavlopoulou, C. Bouyioukos, E. Markou, and P. G. Bagos, "Bipartite graphs in systems biology and medicine: a survey of methods and applications," *Gigascience*, vol. 7, no. 4, p. giy014, 2018.

[3] X. Zhou, J. Menche, A.-L. Barabási, and A. Sharma, "Human symptoms–disease network," *Nature communications*, vol. 5, no. 1, pp. 1–10, 2014.

[4] Y. Li, M. T. Thai, Z. Liu, and W. Wu, "Protein-protein interaction and group testing in bipartite graphs," *International journal of bioinformatics research and applications*, vol. 1, no. 4, pp. 414–419, 2005.

[5] C. Aggarwal, *Managing and Mining Uncertain Data*, 01 2009, vol. 35.

[6] B. Yang, D. Wen, L. Qin, Y. Zhang, L. Chang, and R. Li, "Index-based optimal algorithm for computing k-cores in large uncertain graphs," in *ICDE*, 2019.

[7] D. Qiangqiang, L. Rong-hua, W. Guoren, M. Rui, Z. Zhiwei, and Y. Ye, "Core decompostion on uncertain graphs revisited," in *TKDE*, 2021.

[8] B. Francesco, G. Francesco, K. Andreas, and V. Yana, "Core decomposition of uncertain graphs," in *KDD*, 2014.

[9] H. Xin, L. Wei, and L. V. Lakshmanan, "Truss decomposition of probabilistic graphs: Semantics and algorithms," pp. 77–90, 2016.

[10] Z. Zhaonian and Z. Rong, "Truss decomposition of uncertain graphs," *knowl. Inf. Syst*, vol. 50, pp. 197–230, 2017.

[11] Z. Zhaonian, L. Jianzhong, G. Hong, and S. Zhang, "Finding top-k maximal cliques in an uncertain graph," 2010.

[12] M. ArkoProvo, X. Pan, and T. Srikanta, "Mining maximal cliques from an uncertain graph," pp. 243–254, 2015.

[13] L. Rong-Hua, D. Qiangqiang, W. Guoren, M. Zhong, Q. Lu, and Y. Jeffrey, Xu, "Improved algorithms for maximal clique search in uncertain networks," pp. 1178–1189, 2019.

[14] D. Ding, H. Li, Z. Huang, and N. Mamoulis, "Efficient fault-tolerant group recommendation using alpha-beta-core," in *CIKM*. ACM, 2017.

[15] B. Liu, L. Yuan, X. Lin, L. Qin, W. Zhang, and J. Zhou, "Efficient (α, β)-core computation: an index-based approach," in *WWW*, 2019.

[16] Z. Zou, "Bitruss decomposition of bipartite graphs," in *DASFAA*. Springer, 2016.

[17] K. Wang, X. Lin, L. Qin, W. Zhang, and Y. Zhang, "Efficient bitruss decomposition for large-scale bipartite graphs," in *ICDE*. IEEE, 2020.

[18] B. Lyu, L. Qin, X. Lin, Y. Zhang, Z. Qian, and J. Zhou, "Maximum biclique search at billion scale." *PVLDB*, vol. 13, no. 9, 2020.

[19] S. Lehmann, M. Schwartz, and L. K. Hansen, "Biclique communities," *Physical review E*, vol. 78, no. 1, p. 016108, 2008.

[20] J. R. Larsen, M. R. Martin, J. D. Martin, P. Kuhn, and J. B. Hicks, "Modeling the onset of symptoms of covid-19," *Frontiers in public health*, vol. 8, p. 473, 2020.

[21] K. Wang, W. Zhang, X. Lin, Y. Zhang, L. Qin, and Y. Zhang, "Efficient and effective community search on large-scale bipartite graphs," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 85–96.

[22] J. Li, L. Wu, R. Hong, K. Zhang, Y. Ge, and Y. Li, "A joint neural model for user behavior prediction on social networking platforms," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 11, no. 6, pp. 1–25, 2020.

[23] G. Zhou, N. Mou, Y. Fan, Q. Pi, W. Bian, C. Zhou, X. Zhu, and K. Gai, "Deep interest evolution network for click-through rate prediction," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 5941–5948.

[24] A. Beutel, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos, "Copycatch: stopping group attacks by spotting lockstep behavior in social networks," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 119–130.

[25] A. P. Thurow, C. W. Abdalla, J. Younglove-Webb, and B. Gray, "The dynamics of multidisciplinary research teams in academia," *The review of higher education*, vol. 22, no. 4, pp. 425–440, 1999.

[26] L. Danghao, L. Hio, H. Zhipeng, and M. Nikos, "Efficient fault-tolerant group recommendation using α-β-core," pp. 2047–2050, 2017.

[27] P. Michalis, B. Francesco, G. Aristides, and K. George, "k-nearest neighbors in uncertain graphs," *National security agency technical report*, vol. 3, pp. 997–1008, 2010.

[28] L. Rong-Hua, Y. Jeffrey Xu, M. Rui, and J. Tan, "Recursive stratified sampling: A new framework for query evaluation on uncertain graphs," vol. 28, no. 2, pp. 468–482, 2016.

[29] Y. Guo, X. Zhang, F. Esfahani, V. Srinivasan, A. Thomo, and L. Xing, "Multi-stage graph peeling algorithm for probabilistic core decomposition," *arXiv preprint arXiv:2108.06094*, 2021.

[30] Y. Peng, Y. Zhang, W. Zhang, X. Lin, and L. Qin, "Efficient probabilistic k-core computation on uncertain graphs," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 2018, pp. 1192–1203.

[31] E. Fatemeh, D. Mahsa, S. Venkatesh, T. Alex, and W. Kui, "Truss decomposition on large probabilistic networks using h-index," *SSDBM*, p. 145–156, 2021.

[32] Y. He, K. Wang, W. Zhang, X. Lin, and Y. Zhang, "Exploring cohesive subgraphs with vertex engagement and tie strength in bipartite graphs," *Information Sciences*, vol. 572, pp. 277–296, 2021.

[33] Y. Zhang, K. Wang, W. Zhang, X. Lin, and Y. Zhang, "Pareto-optimal community search on large bipartite graphs," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 2647–2656.

[34] K. Wang, X. Lin, L. Qin, W. Zhang, and Y. Zhang, "Vertex priority based butterfly counting for large-scale bipartite networks." *PVLDB*, 2019.

[35] A. E. Sarıyüce and A. Pinar, "Peeling bipartite networks for dense subgraph discovery," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 2018, pp. 504–512.

[36] A. Abidi, L. Chen, R. Zhou, and C. Liu, "Searching personalized k-wing in large and dynamic bipartite graphs," *arXiv preprint arXiv:2101.00810*, 2021.

[37] Y. Zhang, C. A. Phillips, G. L. Rogers, E. J. Baker, E. J. Chesler, and M. A. Langston, "On finding bicliques in bipartite graphs: a novel algorithm and its application to the integration of diverse biological data types," *BMC Bioinform.*, vol. 15, p. 110, 2014. [Online]. Available: https://doi.org/10.1186/1471-2105-15-110

[38] A. Das and S. Tirthapura, "Incremental maintenance of maximal bicliques in a dynamic bipartite graph," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 4, no. 3, pp. 231–242, 2018.

[39] L. Chen, C. Liu, R. Zhou, J. Xu, and J. Li, "Efficient exact algorithms for maximum balanced biclique search in bipartite graphs," in *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, G. Li, Z. Li, S. Idreos, and D. Srivastava, Eds. ACM, 2021, pp. 248–260. [Online]. Available: https://doi.org/10.1145/3448016.3459241